Multivalued Dependencies and a 4NF for XML

Millist W. Vincent and Jixue Liu

School of Computer and Information Science
University of South Australia
{millist.vincent, jixue.liu}@unisa.edu.au

Abstract. While providing syntactic flexibility, XML provides little semantic content and so the study of integrity constraints in XML plays an important role in helping to improve the semantic expressiveness of XML. Functional dependencies (FDs) and multivalued dependencies (MVDs) play a fundamental role in relational databases where they provide semantics for the data and at the same time are the foundation for database design. Since XML documents are closely coupled with relational databases in that XML documents are typically exported and imported from relational databases, the study of FDs and MVDs in XML is of fundamental significance in XML research. In this paper we define multivalued dependencies in XML (XMVDs). We then propose a normal form for XML documents in the presence of XMVDs and justify our normal form by showing that it ensures the elimination of redundancy.

1 Introduction

XML has recently emerged as a standard for data representation and interchange on the Internet [22,1]. While providing syntactic flexibility, XML provides little semantic content and as a result several papers have addressed the topic of how to improve the semantic expressiveness of XML. Among the most important of these approaches has been that of defining integrity constraints in XML [5]. Several different classes of integrity constraints for XML have been defined including key constraints [5,6], path constraints [8], and inclusion constraints [9] and properties such as axiomatization and satisfiability have been investigated for these constraints. However, one topic that has been identified as an open problem in XML research [22] and which has been little investigated is how to extended the traditional integrity constraints in relational databases, namely *functional dependencies* (FDs) and *multivalued dependencies* (MVDs), to XML and then how to develop a normalisation theory for XML. This problem is not of just theoretical interest. The theory of normalisation forms the cornerstone of practical relational database design and the development of a similar theory for XML will similarly lay the foundation for understanding how to design XML documents. In addition, the study of FDs and MVDs in XML is important because of the close connection between XML and relational databases. With current technology, the source of XML data is typically a relational database [1] and relational databases are also normally used to store XML data [13]. Hence, given that FDs

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 14–29, 2003.

and MVDs are the most important constraints in relational databases, the study of these constraints in XML assumes heightened importance over other types of constraints which are unique to XML [7]. The only papers that have specifically addressed the problem of FDs in XML are the recent papers [3,19,20]. Before presenting the contributions of [3,19,20], we briefly outline the approaches to defining FD satisfaction in incomplete relational databases.

There are two approaches, the first called the *weak satisfaction* approach and the other called the *strong satisfaction* approach [4]. In the weak satisfaction approach, a relation is defined to weakly satisfy a FD if there exists *at least one* completion of the relation, obtained by replacing all occurrences of nulls by data values, which satisfies the FD. A relation is said to strongly satisfy a FD if *every* completion of the relation satisfies the FD. Both approaches have their advantages and disadvantages (a more complete discussion of this issue can be found in [19]). The weak satisfaction approach has the advantage of allowing a high degree of uncertainty to be represented in a database but at the expense of making maintenance of integrity constraints much more difficult. In contrast, the strong satisfaction approach restricts the amount of uncertainty that can be represented in a database but makes the maintenance of integrity constraints much easier. However, as argued in [11], both approaches have their place in real world applications and should be viewed as complementary rather than competing approaches. Also, it is possible to combine the two approaches by having some FDs in a relation strongly satisfied and others weakly satisfied [10].

The contribution of [3] was, for the first time, to define FDs in XML (what we call XFDs) and then to define a normal form for an XML document based on the definition of an XFD. However, there are some difficulties with the definition of an XFD given in [3]. The most fundamental problem is that although it is explicitly recognised in the definitions that XML documents have missing information, the definitions in [3], while having some elements of the weak instance approach, are not a strict extension of this approach since there are XFDs that are violated according to the definition in [3] yet there are completions of the tree that satisfy the XFDs (see [19] for an example). As a result, it is not clear that there is any correspondence between weak satisfaction of FDs in incomplete relations and XFD satisfaction in XML documents as defined in [3].

In [19,20] a different and more straightforward approach was taken to defining XFDs. The definition in [19,20] is based on extending the strong satisfaction approach to XML. The definition of an XFD given in [19] was justified formally by two main results. The first result showed that for a very general class of mappings from an incomplete relation into a XML document, a relation strongly satisfies a unary FD (only one attribute on the l.h.s. of the FD) if and only if the corresponding XML document strongly satisfies the corresponding XFD. The second result showed that an XML document strongly satisfies an XFD if and only if every completion of the XML document also satisfies the XFD. The other contributions in [19] were firstly to define a set of axioms for reasoning about the implication of XFDs and show that the axioms are sound for arbitrary XFDs. The final contribution was to define a normal form, based on a modification of

the one proposed in [3], and prove that it is a necessary and sufficient condition for the elimination of redundancy in an XML document.

In this paper we extend the work in [19] and investigate the issue of multi-valued dependencies and normalisation in XML, a topic which to the best of our knowledge has not been investigated previously. We firstly give a definition of MVDs in XML (what we call XMVDs) using an extension of the approach used in [19]. We note that in an allied paper [21], we formally justify the definition by proving that, for a very general class of mappings from relations to XML, a relation satisfies a MVD if and only if the corresponding XML document satisfies the corresponding XMVD. Thus there is a natural correspondence between MVDs in relations and XMVDs in XML documents. We then propose a normal form for XML documents in the presence of XMVDs. We then justify our normal form by showing that it ensures the elimination of redundancy. Finally, we note that in contrast to [19], in the present paper we assume that XML documents do not have missing information and leave the problem of how to extend the approach to the case of missing information for future research.

The rest of this paper is organised as follows. Section 2 contains some preliminary definitions. Section 3 contains the definition of an XMVD. In Section 4 we define a 4NF for XML and prove that it eliminates redundancy. Finally, Section 5 contains some concluding comments.

2 Preliminary Definitions

In this section we present some preliminary definitions that we need before defining XFDs. We model an XML document as a tree as follows.

Definition 1. *Assume a countably infinite set* $\mathbf{E}$ *of element labels (tags), a countable infinite set* $\mathbf{A}$ *of attribute names and a symbol* $\mathcal{S}$ *indicating text. An* XML tree *is defined to be* $T = (V, lab, ele, att, val, v_r)$ *where* V *is a* finite *set of nodes in* T*;* lab *is a function from* V *to* $\mathbf{E} \cup \mathbf{A} \cup \{\mathcal{S}\}$*;* ele *is a partial function from* V *to a sequence of* V *nodes such that for any* $v \in V$*, if* $ele(v)$ *is defined then* $lab(v) \in \mathbf{E}$*;* att *is a partial function from* $V \times \mathbf{A}$ *to* V *such that for any* $v \in V$ *and* $l \in \mathbf{A}$*, if* $att(v, l) = v_1$ *then* $lab(v) \in \mathbf{E}$ *and* $lab(v_1) = l$*;* val *is a function such that for any node in* $v \in V, val(v) = v$ *if* $lab(v) \in \mathbf{E}$ *and* $val(v)$ *is a string if either* $lab(v) = \mathcal{S}$ *or* $lab(v) \in \mathbf{A}$*;* v_r *is a distinguished node in* V *called the root of* T *and we define* $lab(v_r) = root$*. Since node identifiers are unique, a consequence of the definition of* val *is that if* $v_1 \in \mathbf{E}$ *and* $v_2 \in \mathbf{E}$ *and* $v_1 \neq v_2$ *then* $val(v_1) \neq val(v_2)$*. We also extend the definition of* val *to sets of nodes and if* $V_1 \subseteq V$*, then* $val(V_1)$ *is the set defined by* $val(V_1) = \{val(v) | v \in V_1\}$*.*

For any $v \in V$*, if* $ele(v)$ *is defined then the nodes in* $ele(v)$ *are called* subelements *of* v*. For any* $l \in \mathbf{A}$*, if* $att(v, l) = v_1$ *then* v_1 *is called an* attribute *of* v*. Note that an XML tree* T *must be a tree. Since* T *is a tree the set of ancestors of a node* v*, is denoted by* $Ancestor(v)$*. The children of a node* v *are also defined as in Definition 1 and we denote the parent of a node* v *by* $Parent(v)$*.*

We note that our definition of *val* differs slightly from that in [6] since we have extended the definition of the *val* function so that it is also defined on element nodes. The reason for this is that we want to include in our definition paths that do not end at leaf nodes, and when we do this we want to compare element nodes by node identity, i.e. node equality, but when we compare attribute or text nodes we want to compare them by their contents, i.e. value equality. This point will become clearer in the examples and definitions that follow.

We now give some preliminary definitions related to paths.

Definition 2. *A* path *is an expression of the form* $l_1.\cdots.l_n$, $n \geq 1$, *where* $l_i \in \mathbf{E} \cup \mathbf{A} \cup \{\mathcal{S}\}$ *for all* $i, 1 \leq i \leq n$ *and* $l_1 = root$. *If* p *is the path* $l_1.\cdots.l_n$ *then* $Last(p) = l_n$.

For instance, if $\mathbf{E}$ = {`root, Division, Employee`} and $\mathbf{A}$ = {`D#, Emp#`} then `root, root.Division, root.Division.D#,`
`root.Division.Employee.Emp#.S` are all paths.

Definition 3. *Let* p *denote the path* $l_1.\cdots.l_n$. *The function* $Parnt(p)$ *is the path* $l_1.\cdots.l_{n-1}$. *Let* p *denote the path* $l_1.\cdots.l_n$ *and let* q *denote the path* $q_1.\cdots.q_m$. *The path* p *is said to be a* prefix *of the path* q, *denoted by* $p \subseteq q$, *if* $n \leq m$ *and* $l_1 = q_1, \ldots, l_n = q_n$. *Two paths* p *and* q *are equal, denoted by* $p = q$, *if* p *is a prefix of* q *and* q *is a prefix of* p. *The path* p *is said to be a* strict prefix *of* q, *denoted by* $p \subset q$, *if* p *is a prefix of* q *and* $p \neq q$. *We also define the intersection of two paths* p_1 *and* p_2, *denoted but* $p_1 \cap p_2$, *to be the maximal common prefix of both paths. It is clear that the intersection of two paths is also a path.*

For example, if $\mathbf{E}$ = {`root, Division, Employee`} and $\mathbf{A}$ = {`D#, Emp#`} then `root.Division` is a strict prefix of `root.Division.Employee` and `root.Division.D#` ∩ `root.Division.Employee.Emp#.S` = `root.Division`.

Definition 4. *A* path instance *in an XML tree* T *is a sequence* $\bar{v}_1.\cdots.\bar{v}_n$ *such that* $\bar{v}_1 = v_r$ *and for all* $\bar{v}_i, 1 < i \leq n, v_i \in V$ *and* $\bar{v}_i$ *is a child of* $\bar{v}_{i-1}$. *A path instance* $\bar{v}_1.\cdots.\bar{v}_n$ *is said to be* defined over the path $l_1.\cdots.l_n$ *if for all* $\bar{v}_i, 1 \leq i \leq n$, $lab(\bar{v}_i) = l_i$. *Two path instances* $\bar{v}_1.\cdots.\bar{v}_n$ *and* $\bar{v}'_1.\cdots.\bar{v}'_n$ *are said to be* distinct *if* $v_i \neq v'_i$ *for some* i, $1 \leq i \leq n$. *The path instance* $\bar{v}_1.\cdots.\bar{v}_n$ *is said to be a* prefix *of* $\bar{v}'_1.\cdots.\bar{v}'_m$ *if* $n \leq m$ *and* $\bar{v}_i = \bar{v}'_i$ *for all* $i, 1 \leq i \leq n$. *The path instance* $\bar{v}_1.\cdots.\bar{v}_n$ *is said to be a* strict prefix *of* $\bar{v}'_1.\cdots.\bar{v}'_m$ *if* $n < m$ *and* $\bar{v}_i = \bar{v}'_i$ *for all* $i, 1 \leq i \leq n$. *The set of path instances over a path* p *in a tree* T *is denoted by* $Paths(p)$

For example, in Figure 1, $v_r.v_1.v_3$ is a path instance defined over the path `root.Dept.Section` and $v_r.v_1.v_3$ is a strict prefix of $v_r.v_1.v_3.v_4$

We now assume the existence of a set of legal paths P for an XML application. Essentially, P defines the semantics of an XML application in the same way that a set of relational schema define the semantics of a relational application. P may be derived from the DTD, if one exists, or P be derived from some other source which understands the semantics of the application if no DTD exists. The advantage of assuming the existence of a set of paths, rather than a DTD, is that

it allows for a greater degree of generality since having an XML tree conforming to a set of paths is much less restrictive than having it conform to a DTD. Firstly we place the following restriction on the set of paths.

Definition 5. *A set P of paths is* consistent *if for any path $p \in P$, if $p_1 \subset p$ then $p_1 \in P$.*

This is natural restriction on the set of paths and any set of paths that is generated from a DTD will be consistent.

We now define the notion of an XML tree conforming to a set of paths P.

Definition 6. *Let P be a consistent set of paths and let T be an XML tree. Then T is said to* conform *to P if every path instance in T is a path instance over some path in P.*

The next issue that arises in developing the machinery to define XFDs is the issue is that of missing information. This is addressed in [19] but in this we take the simplifying assumption that there is no missing information in XML trees. More formally, we have the following definition.

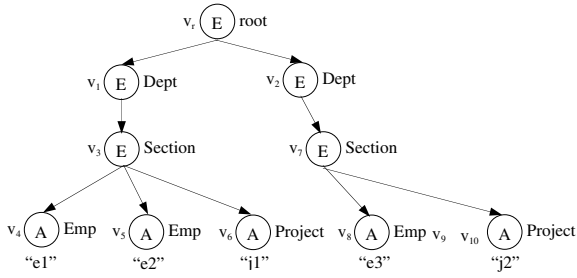

Fig. 1. A complete XML tree.

Definition 7. *Let P be a consistent set of paths, let T be an XML that conforms to P. Then T is defined to be* complete *if whenever there exist paths p_1 and p_2 in P such that $p_1 \subset p_2$ and there exists a path instance $\bar{v}_1.\cdots.\bar{v}_n$ defined over p_1, in T, then there exists a path instance $\bar{v}'_1.\cdots.\bar{v}'_m$ defined over p_2 in T such that $\bar{v}_1.\cdots.\bar{v}_n$ is a prefix of the instance $\bar{v}'_1.\cdots.\bar{v}'_m$.*

For example, if we take P to be {`root, root.Dept, root.Dept.Section, root.Dept.Section.Emp, root.Dept.Section.Project`} then the tree in Figure 1 conforms to P and is complete.

The next function returns all the final nodes of the path instances of a path p in T.

Definition 8. *Let P be a consistent set of paths, let T be an XML tree that conforms to P . The function $N(p)$, where $p \in P$, is the set of nodes defined by $N(p) = \{\bar{v}|\bar{v}_1.\cdots.\bar{v}_n \in Paths(p) \wedge \bar{v} = \bar{v}_n\}$.*

For example, in Figure 1, $N(\texttt{root.Dept}) = \{v_1, v_2\}$.
We now need to define a function that returns a node and its ancestors.

Definition 9. *Let P be a consistent set of paths, let T be an XML tree that conforms to P. The function $AAncestor(v)$, where $v \in V \cup \mathbf{N}$, is the set of nodes in T defined by $AAncestor(v) = v \cup Ancestor(v)$.*

For example in Figure 1, $AAncestor(v_3) = \{v_r, v_1, v_3\}$. The next function returns all nodes that are the final nodes of path instances of p and are descendants of v.

Definition 10. *Let P be a consistent set of paths, let T be an XML tree that conforms to P. The function $Nodes(v, p)$, where $v \in V \cup \mathbf{N}$ and $p \in P$, is the set of nodes in T defined by $Nodes(v, p) = \{x|x \in N(p) \wedge v \in AAncestor(x)\}$*

For example, in Figure 1 , $Nodes(v_1, \texttt{root.Dept.Section.Emp}) = \{v_4, v_5\}$.
We also define a partial ordering on the set of nodes as follows.

Definition 11. *The partial ordering $>$ on the set of nodes V in an XML tree T is defined by $v_1 > v_2$ iff $v_2 \in Ancestor(v_1)$.*

3 XMVDs in XML

Before presenting the main definition of the paper, we present an example to illustrate the thinking behind the definition. Consider the relation shown in Figure 2. It satisfies the MVD $\texttt{Course} \rightarrow\rightarrow \texttt{Teacher|Text}$. The XML tree shown in Figure 3 is then a XML representation of the data in Figure 2. The tree has the following property. There exists two path instances of `root.Id.Id.Id.Text`, namely $v_r.v_{13}.v_{17}.v_{21}.v_9$ and $v_r.v_{16}.v_{20}.v_{24}.v_{12}$ such that $val(v_9) \neq val(v_{12})$. Also, these two paths have the property that for the closest `Teacher` node to v_9, namely v_5, and the closest `Teacher` node to v_{12}, namely v_8, then $val(v_5) \neq val(v_8)$ and for the closest `Course` node to both v_9 and v_5, namely v_1, and for the closest `Course` node to both v_{12} and v_8, namely v_4, we have that $val(v_1) = val(v_4)$. Then the existence of the two path instances $v_r.v_{13}.v_{17}.v_{21}.v_9$ and $v_r.v_{16}.v_{20}.v_{24}.v_{12}$ with these properties and the fact that $\texttt{Course} \rightarrow\rightarrow \texttt{Teacher|Text}$ is satisfied in the relation in Figure 2 implies that there exists two path instances of `root.Id.Id.Id.Text`, namely $v_r.v_{15}.v_{19}.v_{23}.v_{11}$ and $v_r.v_{14}.v_{18}.v_{22}.v_{10}$, with the following properties. $val(v_{11}) = val(v_9)$ and for the closest `Teacher` node to v_{11}, v_7, $val(v_7) = val(v_8)$ and for the closest `Course` node to v_{11} and v_7, namely v_3, $val(v_3) = val(v_1)$. Also, $val(v_{10}) = val(v_{12})$ and the closest `Teacher` node to v_{10}, v_6, $val(v_6) = val(v_5)$ and for the closest `Course` node to v_{10} and v_6, namely v_2, $val(v_2) = val(v_4)$. This type of constraint is an XMVD. We note however that there are many other ways that the relation in Figure 2 could be represented in

an XML tree. For instance we could also represent the relation by Figure 4 and this XML tree also satisfies the XMVD. In comparing the two representations, it is clear that the representation in Figure 4 is a more compact representation than that in Figure 3 and we shall see later that the example in Figure 4 is normalised whereas the example in Figure 3 is not.

Course	Teacher	Text
Algorithms	Fred	Text A
Algorithms	Mary	Text B
Algorithms	Fred	Text B
Algorithms	Mary	Text A

Fig. 2. A flat relation satisfying a MVD.

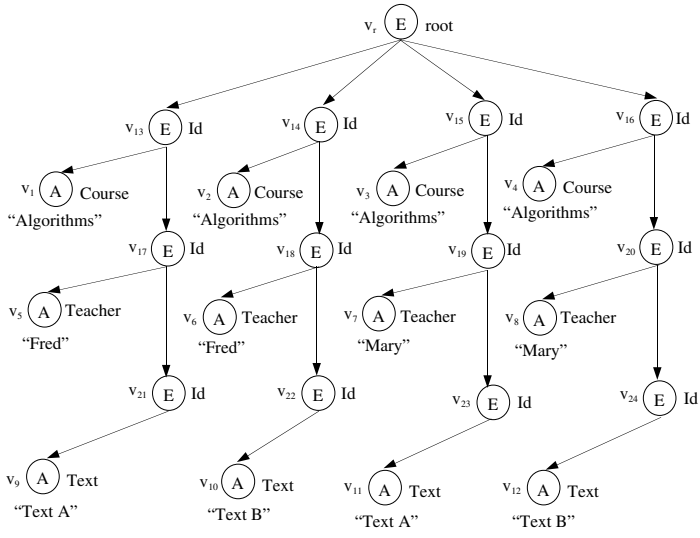

Fig. 3. An XML tree

This leads us to the main definition of our paper. In this paper we consider the simplest case where there are only single paths on the l.h.s. and r.h.s. of the XMVD and all paths end in an attribute or text node.

Definition 12. *Let P be a consistent set of paths and let T be an XML tree that conforms to P and is complete. An XMVD is a statement of the form $p \rightarrow\rightarrow q|r$ where p, q and r are paths in P. T satisfies $p \rightarrow\rightarrow q|r$ if whenever there exists two distinct paths path instances $\bar{v}_1.\cdots.\bar{v}_n$ and $\bar{w}_1.\cdots.\bar{w}_n$ in $Paths(q)$ such that:*

(i) $val(\bar{v}_n) \neq val(\bar{w}_n)$;

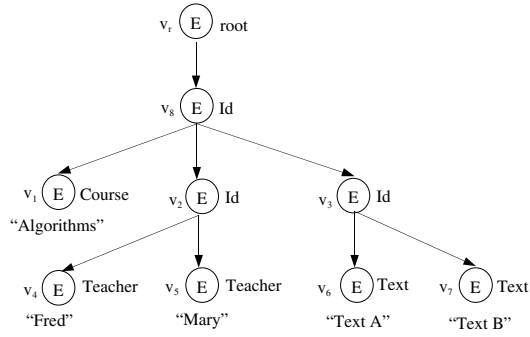

Fig. 4. An XML tree

(ii) there exists two nodes z_1, z_2*, where* $z_1 \in Nodes(x_{1_1}, r)$ *and* $z_2 \in Nodes(y_{1_1}, r)$ *such that* $val(z_1) \neq val(z_2)$*;*

(iii) there exists two nodes z_3 *and* z_4*, where* $z_3 \in Nodes(x_{1_{1_1}}, p)$ *and* $z_4 \in Nodes(y_{1_{1_1}}, p)$*, such that* $val(z_3) = val(z_4)$*;*

then:

(a) there exists a path $\bar{v}'_1.\cdots.\bar{v}'_n$ *in* $Paths(q)$ *such that* $val(\bar{v}'_n) = val(\bar{v}_n)$ *and there exists a node* z'_1 *in* $Nodes(x'_{1_1}, r)$ *such that* $val(z'_1) = val(z_2)$ *and there exists a node* z'_3 *in* $Nodes(x'_{1_{1_1}}, p)$ *such that* $val(z'_3) = val(z_3)$*;*

(b) there exists a path $\bar{w}'_1.\cdots.\bar{w}'_n$ *in* $Paths(q)$ *such that* $val(\bar{w}'_n) = val(\bar{w}_n)$ *and there exists a node* z'_2 *in* $Nodes(x'_{1_1}, r)$ *such that* $val(z'_2) = val(z_1)$ *and there exists a node* z'_4 *in* $Nodes(x'_{1_{1_1}}, p_l)$ *such that* $val(z'_4) = val(z_4)$*;*

where $x_{1_1} = \{v | v \in \{\bar{v}_1, \cdots, \bar{v}_n\} \wedge v \in N(r \cap q)\}$ *and* $y_{1_1} = \{v | v \in \{\bar{w}_1, \cdots, \bar{v}_n\} \wedge v \in N(r \cap q)\}$ *and* $x_{1_{1_1}} = \{v | v \in \{\bar{v}_1, \cdots, \bar{v}_n\} \wedge v \in N(p \cap r \cap q)\}$ *and* $y_{1_{1_1}} = \{v | v \in \{\bar{w}_1, \cdots, \bar{w}_n\} \wedge v \in N(p \cap r \cap q)\}$

and $x'_{1_1} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(r \cap q)\}$ *and* $y'_{1_1} = \{v | v \in \{\bar{w}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(r \cap q)\}$ *and* $x'_{1_{1_1}} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(p \cap r \cap q)\}$ *and* $y'_{1_{1_1}} = \{v | v \in \{\bar{w}'_1, \cdots, \bar{w}'_n\} \wedge v \in N(p \cap r \cap q)\}$.

We note that since the path $r \cap q$ is a prefix of q, there exists only one node in $\bar{v}_1.\cdots.\bar{v}_n$ that is also in $N(r \cap q)$ and so x_1 is always defined and is a single node. Similarly for $y_1, x_{1_{1_1}}, y_{1_{1_1}}, x'_{1_1}, y'_{1_1}, x'_{1_{1_1}}, y'_{1_{1_1}}$. We now illustrate the definition by some examples.

Example 1. Consider the XML tree shown in Figure 4 and the XMVD `root.Id.Course` $\twoheadrightarrow$ `root.Id.Id.Teacher|root.Id.Id.Text`. Let $\bar{v}_1.\cdots.\bar{v}_n$ be the path instance $v_r.v_8.v_2.v_4$ and let $\bar{w}_1.\cdots.\bar{w}_n$ be the path instance $v_r.v_8.v_2.v_5$. Both path instances are in $Paths$(`root.Id.Id.Teacher`) and $val(v_4) \neq val(v_5)$. Moreover, $x_{1_1} = v_8$, $y_{1_1} = v_8$, $x_{1_{1_1}} = v_8$ and $y_{1_{1_1}} = v_8$. So if we let $z_1 = v_6$ and $z_2 = v_7$ then $z_1 \in Nodes(x_{1_1}$, `root.Id.Id.Text`) and

$z_2 \in Nodes(y_{1_1}$, `root.Id.Id.Text`). Also if we let $z_3 = v_8$ and $z_4 = v_8$ then $z_3 \in Nodes(x_{1_{1_1}}$, `root.Id.Course`) and $z_4 \in Nodes(y_{1_{1_1}}$, `root.Id.Course`) then $val(z_3) = val(z_4)$. Hence conditions (i), (ii) and (iii) of the definition of an XMVD are satisfied.

If we let $\bar{v}'^{i}_{1}.\cdots.\bar{v}'^{i}_{n}$ be the path $v_r.v_8.v_2.v_4$ we firstly have that $val(\bar{v}'^{i}_{n}) = val(\bar{v}^{i}_{n})$ as required. Also, since the path instances are the same we have that $x_{1_1} = x'_{1_1}$ and $x_{1_{1_1}} = x'_{1_{1_1}}$. So if we let $z'_1 = v_7$ then

$z'_1 \in Nodes(x'_{1_1}, \texttt{root.Id.Id.Text})$ and $val(z'_1) = val(z_2)$ and if we let $z'_3 = v_8$ then

$z'_3 \in Nodes(x'_{1_{1_l}}, \texttt{root.Id.Course})$ and $val(z'_3) = val(z_3)$. So part (a) of the definition of an XMVD is satisfied. Next if we let $\bar{w}'^{i}_{1}.\cdots.\bar{w}'^{i}_{n}$ be the path $v_r.v_8.v_2.v_5$ then we firstly have that $val(\bar{w}'^{i}_{n}) = val(\bar{w}^{i}_{n})$ since the paths are the same . Also, since the paths are the same we have that $y_{1_1} = y'_{1_1}$ and $y_{1_{1_1}} = y'_{1_{1_1}}$. So if we let $z'_2 = v_6$ then $z'_2 \in Nodes(y'_{1_1}, \texttt{root.Id.Id.Text})$ and $val(z'_2) = val(z_1)$ and if we let $z'_4 = v_8$ then $z'_4 \in Nodes(x'_{1_{1_l}}, \texttt{root.Id.Course})$ and $val(z'_4) = val(z_4)$. Hence part (b) on the definition of an XMVD is satisfied and so T satisfies $\texttt{root.Id.Course} \rightarrow\rightarrow \texttt{root.Id.Id.Teacher|root.Id.Id.Text}$.

As explained earlier, the tree in Figure 4 also satisfies

$\texttt{root.Id.Course} \rightarrow\rightarrow \texttt{root.Id.Id.Teacher|root.Id.Id.Text}$.

Example 2. Consider the XML tree shown in Figure 5 and the XMVD $\texttt{root.Project.P#} \rightarrow\rightarrow \texttt{Root.Project.Person.Name|root.Project.Part.Pid}$. For the path instances $v_r.v_1.v_5.v_{13}$ and $v_r.v_2.v_8.v_{16}$ in

$Paths(\texttt{Root.Project.Person.Name})$ we have that $val(v_{13}) \neq val(v_{16})$. Moreover, $x_{1_1} = v_1$, $y_{1_1} = v_2$, $x_{1_{1_1}} = v_1$ and $y_{1_{1_1}} = v_1$. So if we let $z_1 = v_{17}$ and $z_2 = v_{18}$ then

$z_1 \in Nodes(x_{1_1}, \texttt{root.Project.Part.Pid})$ and

$z_2 \in Nodes(y_{1_1}, \texttt{root.Project.Part.Pid})$. Also if we let $z_3 = v_4$ and $z_4 = v_7$ then $z_3 \in Nodes(x_{1_{1_1}}, \texttt{root.Project.P#})$ and

$z_4 \in Nodes(y_{1_{1_1}}, \texttt{root.Project.P\# })$ and $val(z_3) = val(z_4)$. Hence conditions (i), (ii) and (iii) of the definition of an XMVD are satisfied. However, for the only other path in

$Paths(\texttt{Root.Project.Person.Name})$, namely $v_r.v_3.v_{11}.v_{19}$ we have that $x'_{1_1} = v_3$ and so $Nodes(x'_{1_1}, \texttt{root.Project.part.Pid}) = v_{21}$ and since $val(v_{21}) \neq val(z_2)$ and so it does not satisfy condition (a) and thus $\texttt{root.Project.P#} \rightarrow\rightarrow \texttt{Root.Project.Person.Name|root.Project.part.Pid}$ is violated in T.

Consider then the XMVD XMVD $\texttt{root.Project.Person.Name}$

$\rightarrow\rightarrow \texttt{Root.Project.Person.Skill |root.Project.P\#}$ in the same XML tree. For the path instances $v_r.v_1.v_5.v_{14}$ and $v_r.v_3.v_{11}.v_{20}$ in

$Paths(\texttt{Root.Project.Person.Skill})$ we have that $val(v_{14}) \neq val(v_{20})$. Moreover, $x_{1_1} = v_1$, $y_{1_1} = v_3$, $x_{1_{1_1}} = v_{13}$ and $y_{1_{1_1}} = v_{19}$. So if we let $z_1 = v_4$ and $z_2 = v_{10}$ then $z_1 \in Nodes(x_{1_1}, \texttt{root.Project.P\#})$ and $z_2 \in Nodes(y_{1_1}, \texttt{root.Project.P\#})$. Also if we let $z_3 = v_{13}$ and $z_4 = v_{19}$ then $z_3 \in Nodes(x_{1_{1_1}}, \texttt{root.Project.Person.Name})$ and

$z_4 \in Nodes(y_{1_{1_1}}, \texttt{root.Project.Person.Name})$ and $val(z_3) = val(z_4)$. Hence the conditions of (i), (ii) and (iii) of the definition of an XMVD are satisfied. However there does not exist another path instance in $Paths(\texttt{Root.Project.Person.Skill})$ such that *val* of the last node in the path

is equal to *val* of node v_{14} and so part (a) of the definition of an XMVD is violated.

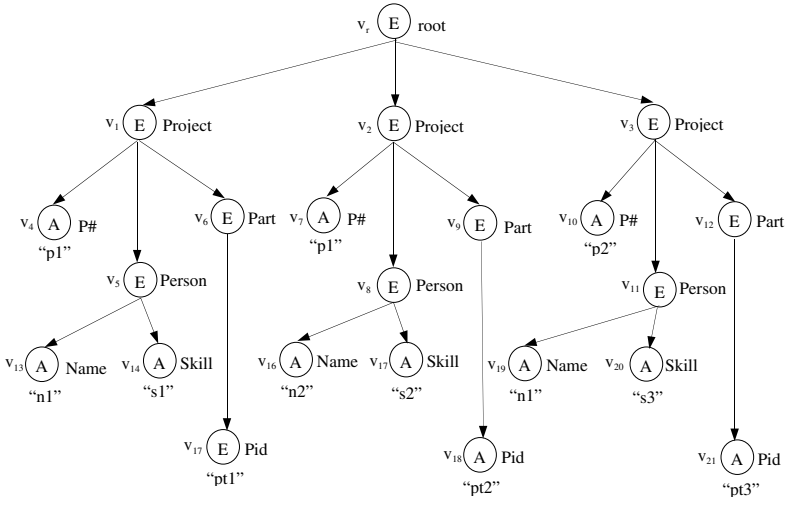

Fig. 5. An XML tree

4 A Redundancy Free 4NF for XML Documents

In this section we propose a 4NF for XML documents. We also provide a formal justification for the normal form by showing that it ensures the elimination of redundancy in the presence of XMVDs. This approach to justifying the definition of a normal form is an extension of the approach adopted by one of the authors in some other research which investigated the issue of providing justification for the normal forms defined in standard relational databases [14,15,16,12].

The approach that we use to justifying our normal form is to formalise the notion of redundancy, the most intuitive approach to justifying normal forms, and then to try to ensure that our normal form ensures there is no redundancy. However, defining redundancy is not quite so straightforward as might first appear. The most obvious approach is, given a relation r and a FD $A \rightarrow B$ and two tuples t_1 and t_2, to define a value $t_1[B]$ to be redundant if $t_1[B] = t_2[B]$ and $t_1[A] = t_2[A]$. While this definition is fine for FDs in relations, it doesn't generalise in an obvious way to other classes of relational integrity constraints, such as *multi-valued dependencies* (MVDs) or *join dependencies* (JDs) or *inclusion dependencies* (INDs), nor to other data models. The key to finding the appropriate generalisation is based on the observation that if a value $t_1[B]$ is redundant in the sense just defined then every change of $t_1[B]$ to a new value results in

the violation of $A \rightarrow B$. One can then define a data value to be redundant if every change of it to a new value results in the violation of the set of constraints (whatever they may be). This is essentially the definition proposed in [16] where it was shown that BCNF is a necessary and sufficient condition for there to be no redundancy in the case of FD constraints and fourth normal form (4NF) is a necessary and sufficient condition for there to be no redundancy in the case of FD and MVD constraints.

The definition we propose is the following which is an extension of the definition given in [16].

Definition 13. *Let T be an XML tree and let v be a node in T. Then the change from v to v', resulting in a new tree T', is said to be a* valid change *if $v \neq v'$ and $val(v) \neq val(v')$.*

We note that the second condition in the definition, $val(v) \neq val(v')$, is automatically satisfied if the first condition is satisfied when $lab(v) \in \mathbf{E}$.

Definition 14. *Let P be a consistent set of paths and let Σ be a set of XMVDs such that every path appearing in an XMVD in Σ is in P. Then Σ is said to* cause redundancy *if there exists a complete XML tree T which conforms to P and satisfies Σ and a node v in T such that* every *valid change from v to v', resulting in a new XML tree T', causes Σ to be violated.*

The essential idea is that if a value is redundant, then it is implied by the other data values and the set of constraints and so any change to the value causes a violation of the constraints. For example, consider Figure 3 and the set Σ of XMVDs

{`root.Id.Course` $\rightarrow\!\!\rightarrow$ `root.Id.Id.Teacher|root.Id.Id.Id.Text`}. Then Σ causes redundancy because the tree shown in Figure 3 satisfies Σ yet every valid change to any of the `Text` nodes (or `Teacher` nodes) results in the violation of Σ.

Before presenting our definition of 4NF we place restrictions on XMVDs associated with an XML document. As shown in the full length version of this paper, these restrictions are not essential but they do simplify and shorten the presentation. The first restriction is on the structure of the XMVDs. In relations and nested relations, MVDs which have a hierarchical structure occur naturally and seem to correspond most closely to real world MVDs [11,17]. Given the close relationship between nested relations and XML documents we believe that the case is similar for XML documents and propose the following definition.

Definition 15. *A set Σ of XMVDs is said to be* hierarchical *if for every XMVD $p \rightarrow\!\!\rightarrow q|r \in \Sigma$, $Parnt(p)$ is a strict prefix of both q and r.*

For instance, in Example 2 the XMVD `root.Project.P#` $\rightarrow\!\!\rightarrow$ `Root.Project.Person.Name|root.Project.Part.Pid` is hierarchical but not the XMVD `root.Project.Person.Name` $\rightarrow\!\!\rightarrow$ `Root.Project.Person.Skill|root.Project.P#`.

Next, we define the notion of a key.

Definition 16. *Let P be a consistent set of paths, let T be an XML tree that conforms to P and is complete and let $p \in P$. Then T satisfies the key constraint p if whenever there exists two nodes v_1 and v_2 in $N(p)$ in T such that $val(v_1) = val(v_2)$ then $v_1 = v_2$.*

This leads us to the normal form definition and the main result of the paper.

Definition 17. *Let Σ be a hierarchical set XMVDs. Then Σ is in* XML fourth normal form *(4XNF) if either of the following conditions hold:*
(A) q and r are both keys;
(B) p is a key and $q \cap r = p$.

We now illustrate the definition by some examples.

Example 3. Consider the tree T in Figure 3 and assume that the only constraint is the XMVD

`root.Id.Course` $\rightarrow\rightarrow$ `root.Id.Id.Teacher|root.Id.Id.Id.Text`. T satisfies Σ and is complete. However Σ is not in 4XNF since `root.Id.Course` is not a key and `root.Id.Id.Teacher` is not a key and `root.Id.Id.Id.Text` is not a key. Consider then the tree shown in Figure 4 and assume that the only XMVD is `root.Id.Course` $\rightarrow\rightarrow$ `root.Id.Teacher|root.Id.Text`. If `root.Id.Course` is a key, which would be the case if `Course` was specified as type `ID` in the full DTD, then Σ is in 4XNF since `root.Id.Course` is a key and `root.Id.Teacher` $\cap$ `root.Id.Id.Id.Text` = textttroot.Id.Course.

This leads to the main result of this section.

Theorem 1. *If Σ is in 4XNF then Σ does not cause redundancy.*

Proof. Assume that (A) holds, i.e. q and r are both keys, and suppose to the contrary that Σ causes redundancy. Then by definition there exists an XML tree T which satisfies Σ and a node v in T such that *every* valid change from v to v', resulting in a new XML tree T', causes some XMVD $p \rightarrow\rightarrow q|r \in \Sigma$ to be violated. We firstly claim that $v \in N(q)$ or $v \in N(r)$. Suppose that this is not the case and that $v \notin N(q)$ and $v \notin N(r)$ and $v \notin N(p)$. Then obviously T' satisfies Σ and so Σ does not cause redundancy which is a contradiction. Suppose then that v in $N(p)$. If $lab(v) \in \mathbf{E}$ then by definition the new node v' is distinct in T' and so by definition of an XMVD T' satisfies Σ and so Σ does not cause redundancy which is a contradiction. If instead $lab(v) \in \mathbf{A}$ then if we choose a change such that $val(v')$ does not appear anywhere else in T, the XMVD $p \rightarrow\rightarrow q|r$ is not violated after the change and so T' satisfies Σ and so Σ does not cause redundancy which is a contradiction. Hence $v \in N(q)$ or $v \in N(r)$. We suppose firstly that $v \in N(q)$.

So since $p \rightarrow\rightarrow q|r$ in T' is violated, there exist path instances $\bar{v}_1.\cdots.\bar{v}_n$ (where $\bar{v}_n = v'$) and $\bar{w}_1.\cdots.\bar{w}_n$ in $Paths(q)$ in T' such that:

(i) $val(\bar{v}_n) \neq val(\bar{w}_n)$;

(ii) there exists two nodes z_1, z_2, where $z_1 \in Nodes(x_{1_1}, r)$ and $z_2 \in Nodes(y_{1_1}, r)$ such that $val(z_1) \neq val(z_2)$;

(iii) there exists two nodes z_3 and z_4, where $z_3 \in Nodes(x_{1_{1_1}}, p)$ and $z_4 \in Nodes(y_{1_{1_1}}, p)$, such that $val(z_3) = val(z_4)$.

and:

(a.1) there does not exist a path $\bar{v}'_1.\cdots.\bar{v}'_n$ in $Paths(q)$ such that $val(\bar{v}'_n) = val(\bar{v}_n)$ and there does not exist a node z'_1 in $Nodes(x'_{1_1}, r)$ such that $val(z'_1) = val(z_2)$ and there does not exist a node z'_3 in $Nodes(x'_{1_{1_1}}, p)$ such that $val(z'_3) = val(z_3)$;

or

(b.1) there does not exists a path $\bar{w}'_1.\cdots.\bar{w}'_n$ in $Paths(q)$ such that $val(\bar{w}'_n) = val(\bar{w}_n)$ and there does not exist a node z'_2 in $Nodes(x'_{1_1}, r)$ such that $val(z'_2) = val(z_1)$ and there does not exist a node z'_4 in $Nodes(x'_{1_{1_1}}, p_l)$ such that $val(z'_4) = val(z_4)$;

where

$x_{1_1} = \{v | v \in \{\bar{v}_1, \cdots, \bar{v}_n\} \wedge v \in N(r \cap q)\}$ and $y_{1_1} = \{v | v \in \{\bar{w}_1, \cdots, \bar{v}_n\} \wedge v \in N(r \cap q)\}$ and $x_{1_{1_1}} = \{v | v \in \{\bar{v}_1, \cdots, \bar{v}_n\} \wedge v \in N(p \cap r \cap q)\}$ and $y_{1_{1_1}} = \{v | v \in \{\bar{w}_1, \cdots, \bar{w}_n\} \wedge v \in N(p \cap r \cap q)\}$

and $x'_{1_1} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(r \cap q)\}$ and $y'_{1_1} = \{v | v \in \{\bar{w}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(r \cap q)\}$ and $x'_{1_{1_1}} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(p \cap r \cap q)\}$ and $y'_{1_{1_1}} = \{v | v \in \{\bar{w}'_1, \cdots, \bar{w}'_n\} \wedge v \in N(p \cap r \cap q)\}$.

Next, because $\bar{v}_1.\cdots.\bar{v}_n$ and $\bar{w}_1.\cdots.\bar{w}_n$ satisfy (ii), it follows that since only node $\bar{v}_n$ is changed in T, then $z_1, z_2, z_3, z_4, x_{1_1}, x'_{1_1}, y_{1_1}, y'_{1_1}, x_{1_{1_1}}, x'_{1_{1_1}}, y_{1_{1_1}}$ and $y'_{1_{1_1}}$ are the same in T and T'. So there exist two nodes z_1, z_2, where $z_1 \in Nodes(x_{1_1}, r)$ and $z_2 \in Nodes(y_{1_1}, r)$ in T such that $val(z_1) \neq val(z_2)$. Also, since q is a key it follows that $val(v) \neq val(\bar{w}_n)$ in T. Consider then the path instances $\bar{v}_1.\cdots.v$ and $\bar{w}_1.\cdots.\bar{w}_n$ in $Paths(q)$ in T. As we have already noted, $val(v) \neq val(\bar{w}_n)$ so (i) of the definition of an XMVD is satisfied. Then since only node v is changed, if we let z_1, z_2 be as defined we have that $z_1 \in Nodes(x_{1_1}, r)$ and $z_2 \in Nodes(y_{1_1}, r)$ and $val(z_1) \neq val(z_2)$ and so (ii) of the definition of an XMVD is satisfied. Similarly, if we let z_3 and z_4 be as defined, then $z_3 \in Nodes(x_{1_{1_1}}, p)$ and $z_4 \in Nodes(y_{1_{1_1}}, p)$ and $val(z_3) = val(z_4)$ and so (iii) of the definition of an XMVD is satisfied. Hence by definition of XMVD satisfaction and since $p \twoheadrightarrow q|r$ is satisfies in T:

(a) there exists a path $\bar{v}'_1.\cdots.\bar{v}'_n$ in $Paths(q)$ in T such that $val(\bar{v}'_n) = val(\bar{v}_n)$ and there exists a node z''_1 in $Nodes(s'_{1_1}, r)$ such that $val(z''_1) = val(z_2)$ and there exists a node z''_3 in $Nodes(s'_{1_{1_1}}, p)$ such that $val(z''_3) = val(z_3)$ where $s'_{1_1} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(r \cap q)\}$ and $s'_{1_{1_1}} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(p \cap r \cap q)\}$;

and

(b) there exists a path $\bar{w}'_1.\cdots.\bar{w}'_n$ in $Paths(q)$ in T such that $val(\bar{w}'_n) = val(\bar{w}_n)$ and there exists a node z''_2 in $Nodes(t'_{1_1}, r)$ such that $val(z''_2) = val(z_1)$ and there exists a node z''_4 in $Nodes(t'_{1_{1_1}}, p_l)$ such that $val(z''_4) = val(z_4)$ where $t'_{1_1} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(r \cap q)\}$ and $t'_{1_{1_1}} = \{v | v \in \{\bar{v}'_1, \cdots, \bar{v}'_n\} \wedge v \in N(p \cap r \cap q)\}$.

We claim that $\bar{v}'_1.\cdots.\bar{v}'_n$ must be distinct from $\bar{v}_1.\cdots.v$. Suppose that it is not. Consider then the result of the change to v. If we let the path $\bar{v}''_1.\cdots.\bar{v}''_n$

in T' be the same as the path instance $\bar{v}_1.\cdots.\bar{v}_n$, and $\bar{w}''_1.\cdots.\bar{w}''_n$ be the same as $\bar{w}_1.\cdots.\bar{w}_n$ then as we have seen $\bar{v}''_1.\cdots.\bar{v}''_n$ and $\bar{w}''_1.\cdots.\bar{w}''_n$ satisfy (i), (ii) and (iii) of the definition of an XMVD. However, if we let $z'_1 = z''_1$ and $z'_3 = z''_3$ then z'_1 in $Nodes(s'_{1_1}, r)$ and $val(z'_1) = val(z_2)$ and z'_3 in $Nodes(s'_{1_{1_1}}, p)$ and $val(z'_3) = val(z_3)$ so (a) of the definition of an XMVD is satisfied. Similarly, if we let $z'_2 = z''_2$ and $z'_4 = z''_4$ then z'_2 in $Nodes(t'_{1_1}, r)$ and $val(z'_2) = val(z_1)$ and z'_4 in $Nodes(t'_{1_{1_1}}, p)$ and $val(z'_4) = val(z_4)$ so (b) of the definition of an XMVD is satisfied. This contradicts the earlier fact that either (a.1) or (b.1) is satisfied since $p \twoheadrightarrow q|r$ is violated in T' and so we conclude that $\bar{v}'_1.\cdots.\bar{v}'_n$ must be distinct from $\bar{v}_1.\cdots.v$. However, by (a) above $val(\bar{v}'_n) = val(\bar{v}_n) = val(v)$ which contradicts the fact that q is a key.

Similarly, if $v \in N(r)$ then using the same arguments we contradict the fact that r is a key and so we conclude that Σ does not cause redundancy if (A) of the definition of 4XNF holds.

Assume next that (B) holds, i.e. p is a key and $q \cap r = p$, and suppose to the contrary that Σ causes redundancy.. Then, as before, there exists an XML tree T which satisfies Σ and a node v in T such that *every* valid change from v to v', resulting in a new XML tree T', causes some XMVD $p \twoheadrightarrow q|r \in \Sigma$ to be violated. Using the same arguments as in (A), it follows that $v \in N(q)$ or $v \in N(r)$. We suppose firstly that $v \in N(q)$. So since $p \twoheadrightarrow q|r$ in T' is violated, there exist path instances $\bar{v}_1.\cdots.\bar{v}_n$ (where $\bar{v}_n = v'$) and $\bar{w}_1.\cdots.\bar{w}_n$ in $Paths(q)$ in T' such that:

(i) $val(\bar{v}_n) \neq val(\bar{w}_n)$;

(ii) there exists two nodes z_1, z_2, where $z_1 \in Nodes(x_{1_1}, r)$ and
$z_2 \in Nodes(y_{1_1}, r)$ such that $val(z_1) \neq val(z_2)$;

(iii) there exists two nodes z_3 and z_4, where $z_3 \in Nodes(x_{1_{1_1}}, p)$ and $z_4 \in Nodes(y_{1_{1_1}}, p)$, such that $val(z_3) = val(z_4)$.

and:

(a.1) there does not exist a path $\bar{v}'_1.\cdots.\bar{v}'_n$ in $Paths(q)$ such that $val(\bar{v}'_n) = val(\bar{v}_n)$ and there does not exist a node z'_1 in $Nodes(x'_{1_1}, r)$ such that $val(z'_1) = val(z_2)$ and there does not exist a node z'_3 in $Nodes(x'_{1_{1_1}}, p)$ such that $val(z'_3) = val(z_3)$;

or

(b.1) there does not exists a path $\bar{w}'_1.\cdots.\bar{w}'_n$ in $Paths(q)$ such that $val(\bar{w}'_n) = val(\bar{w}_n)$ and there does not exist a node z'_2 in $Nodes(x'_{1_1}, r)$ such that $val(z'_2) = val(z_1)$ and there does not exist a node z'_4 in $Nodes(x'_{1_{1_1}}, p_l)$ such that $val(z'_4) = val(z_4)$;

Consider then the paths $\bar{v}_1.\cdots.v$ and $\bar{w}_1.\cdots.\bar{w}_n$ in $Paths(q)$ in T. Since p is a key and $q \cap r = p$, then $x_{1_1} = x'_{1_1} = y_{1_1} = x'_{1_1}$ and $x_{1_{1_1}} = x'_{1_{1_1}} = y_{1_{1_1}} = y'_{1_{1_1}}$. Hence if T' violates $p \twoheadrightarrow q|r$ then so will T which is a contradiction. The same argument applies if $v \in N(r)$ and so we conclude that Σ does not cause redundancy. □

5 Conclusions

In this paper we have investigated the issues of multivalued dependencies and 4NF in XML. We firstly gave a definition of XMVDs in XML using an extension of the approach used in [19,20]. We then proposed a normal form for XML documents in the presence of XMVDs and justified it by showing that it ensures the elimination of redundancy for an important class of XMVDs, namely what we call *hierarchical MVDs*. Hierarchical XMVDs are a natural extension of hierarchical multivalued dependencies that appear in relational and nested relational models.

There are several other issues related to the ones addressed in this paper that warrant further investigation. The first is the need to generalise the main result of this paper. We need to show that 4XNF we proposed is also a necessary condition for the elimination of redundancy and also to extend the definition of 4XNF to the case where the XMVDs are not hierarchical. Secondly, we need to investigate the problem of developing an axiom system for reasoning about the implication of XMVDs. In [19] an axiom system for reasoning about the implication of XFDs was developed and the system was shown to be sound for arbitrary XFDs. Later [18], the axiom system was shown to be complete for unary XFDs and a polynomial time algorithm was developed for determining if a unary XFD is implied by a set of unary XFDs. Similarly, we need to develop an axiom system and algorithm for the implication problem for XMVDs. Thirdly, we need to develop algorithms for converting unnormalised XML documents to normalised ones. In the relational case, the equivalent procedure is performed using a decomposition algorithm based on the projection operator. However, at the moment there has been no commonly agreed upon algebra defined for XML, let alone a projection operator, so the development of procedures for normalising XML documents is likely to be more complex than in the relational case. Fourthly, it is necessary to consider the case where both XFDs and XMVDs exist in a document. It is interesting to note that unlike the situation for the relational case, 4XNF is not a straightforward generalisation of the normal form for XFDs (XNF). This means that, in contrast to the relational case where 4NF implies BCNF in the case where both MVDs and FDs are present, in XML a different normal form from 4XNF is needed when the constraints on an XML document contain both XFDs and XMVDs. The situation is further complicated by the fact that XMVDs and XFDs interact, in the sense that there are XMVDs and XFDs implied by a combined set of XMVDs and XFDs which are not implied by either the XMVDs or XFDs considered alone. This situation parallels that of relational databases [2].

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web.* Morgan Kauffman, 2000.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases.* Addison Wesley, 1996.

3. M. Arenas and L. Libkin. A normal form for xml documents. In *Proc. ACM PODS Conference*, pages 85–96, 2002.
4. P. Atzeni and V. DeAntonellis. *Foundations of databases*. Benjamin Cummings, 1993.
5. P. Buneman, S. Davidson, W. Fan, and C. Hara. Reasoning about keys for xml. In *International Workshop on Database Programming Languages*, 2001.
6. P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Keys for xml. *Computer Networks*, 39(5):473–487, 2002.
7. P. Buneman, W. Fan, J. Simeon, and S. Weinstein. Constraints for semistructured data and xml. *ACM SIGMOD Record*, 30(1):45–47, 2001.
8. P. Buneman, W. Fan, and S. Weinstein. Path constraints on structured and semistructured data. In *Proc. ACM PODS Conference*, pages 129–138, 1998.
9. W. Fan and J. Simeon. Integrity constraints for xml. In *Proc. ACM PODS Conference*, pages 23–34, 2000.
10. M. Levene and G. Loizu. Axiomatization of functional dependencies in incomplete relations. *Theoretical Computer Science*, 206:283–300, 1998.
11. M. Levene and G. Loizu. *A guided tour of relational databases and beyond*. Springer, 1999.
12. M. Levene and M. W. Vincent. Justification for inclusion dependency normal form. *IEEE Transactions on Knowledge and Data Engineering*, 12:281–291, 2000.
13. J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton:. Relational databases for querying xml documents: Limitations and opportunities. In *VLDB Conference*, pages 302–314, 1999.
14. M. W. Vincent. A corrected 5nf definition for relational database design. *Theoretical Computer Science*, 185:379–391, 1997.
15. M. W. Vincent. A new redundancy free normal form for relational database design. In B. Thalheim and L. Libkin, editors, *Database Semantics*, pages 247–264. Springer Verlag, 1998.
16. M. W. Vincent. Semantic foundations of 4nf in relational database design. *Acta Informatica*, 36:1–41, 1999.
17. M. W. Vincent and M. Levene. Restructuring partitioned normal relations without information loss. *SIAM Journal on Computing*, 39(5):1550–1567, 2000.
18. M.W. Vincent and J. Liu. The implication problem for unary functional dependencies in xml. Submitted for publication, 2002.
19. M.W. Vincent and J. Liu. Strong functional dependencies and a redundancy free normal form for xml. Submitted for publication, 2002.
20. M.W. Vincent and J. Liu. Functional dependencies for xml. In *Fifth Asian Pacific Web Conference*, 2003.
21. M.W. Vincent and J. Liu. Multivalued dependencies for xml. In *20th BNCOD Conference*, 2003.
22. J. Widom. Data management for xml - research directions. *IEEE data Engineering Bulletin*, 22(3):44–52, 1999.